
AMBASSADORS
OF CODE

The Laboratory's long history of developing and supporting open-source software has led to thriving user communities and international collaborations.

LAWRENCE Livermore develops and enhances numerical simulation codes to support basic science research, advance next-generation computer science, improve simulation and modeling capabilities, and meet the growing demands of high-performance computing (HPC) systems. Although some of this software is classified or controlled, much of the code developed at Livermore can benefit a range of users and applications outside the Laboratory. An increasingly vital form of technology transfer occurs through the release of open-source software (OSS), wherein the source code is made freely available to the public for inspection, modification, and enhancement.

For decades, the Laboratory has made software developed for programmatic work publicly available as open source. In 2002, the Department of Energy (DOE) issued a software release policy recommending that code produced for its programs be made open source unless exceptions could be justified. Several years later, in 2016, the U.S. government established the Federal Source Code Policy, mandating that code developed by or for government institutions be made available to other federal agencies. The policy further directs government agencies to provide at least 20 percent of their code to the public as OSS. In recent years, Laboratory scientists and engineers have organized unclassified software repositories on mainstream open-source hosting services, such as GitHub, and built active communities with external collaborators. Today, more than 350 software packages developed at Livermore are available to federal, industry, academic, and other public users.

Livermore's OSS addresses several key HPC and non-HPC needs—compilers, package managers, data analytics, visualization tools, input/output benchmarking, data storage, parallel performance, workload management, and math and physics codes—across multiple operating systems and programming languages. Supported by diverse funding streams, all directorates participate in developing valuable software resources. The Laboratory's presence—and reputation—in the open-source community spans online repositories, conferences, publications, and social media.

Strength in User Numbers

Releasing software as open source is a common industry practice. Microsoft and Google, for example, make portions of their software widely available, and all major web browsers and front-end web development languages are built on open-source technologies. Scientists who incorporate OSS into their disciplines have begun to reference and include source codes in scientific papers to encourage reproducibility. The Laboratory's open-source strategy has several advantages that help Livermore keep pace with rapidly growing technological and market needs.

According to Livermore computer engineer Ian Lee, the Federal Source Code Policy acknowledges the value of OSS both among and beyond national laboratories. He explains, "Government agencies and others are trying to solve the same problems. The open-source mandate allows agencies and contractors to align and make the most of external resources. We can share projects and avoid redundancies. In particular, the specialized HPC community benefits from information sharing. Livermore computer scientist Todd Gamblin says, "There aren't many sites that conduct large-scale HPC work, and we need to develop common HPC infrastructure with collaborators rather than perpetuate siloed efforts."

Building a community around an open-source project enables users to

provide valuable feedback, which can result in useful contributions such as new features and bug fixes. In cybersecurity, for example, open-source encryption software is viewed by developers and researchers as more trustworthy than closed-source (proprietary or licensed) software because the former can be scrutinized by the larger user community. Indeed, Laboratory scientists often leverage externally developed software for internal projects and programs (see the box on p. 10). In addition, computational mathematician Tzanio Kolev reasons, "When you know other people will review your work, you impose a higher standard for yourself." The feedback process improves industry standards and Livermore's computing capabilities. Open-source exposure can also prompt discovery of new applications and commercialization potential for a software program. For instance, the Laboratory-developed DYNA3D code used for simulating the mechanical behavior of collision events was released as OSS in 1978. A more advanced commercial code, LS-DYNA, was built on its predecessor's success. LS-DYNA became the leading commercially licensed product for collision event simulation (see *S&TR*, June 2017, pp. 4–11).

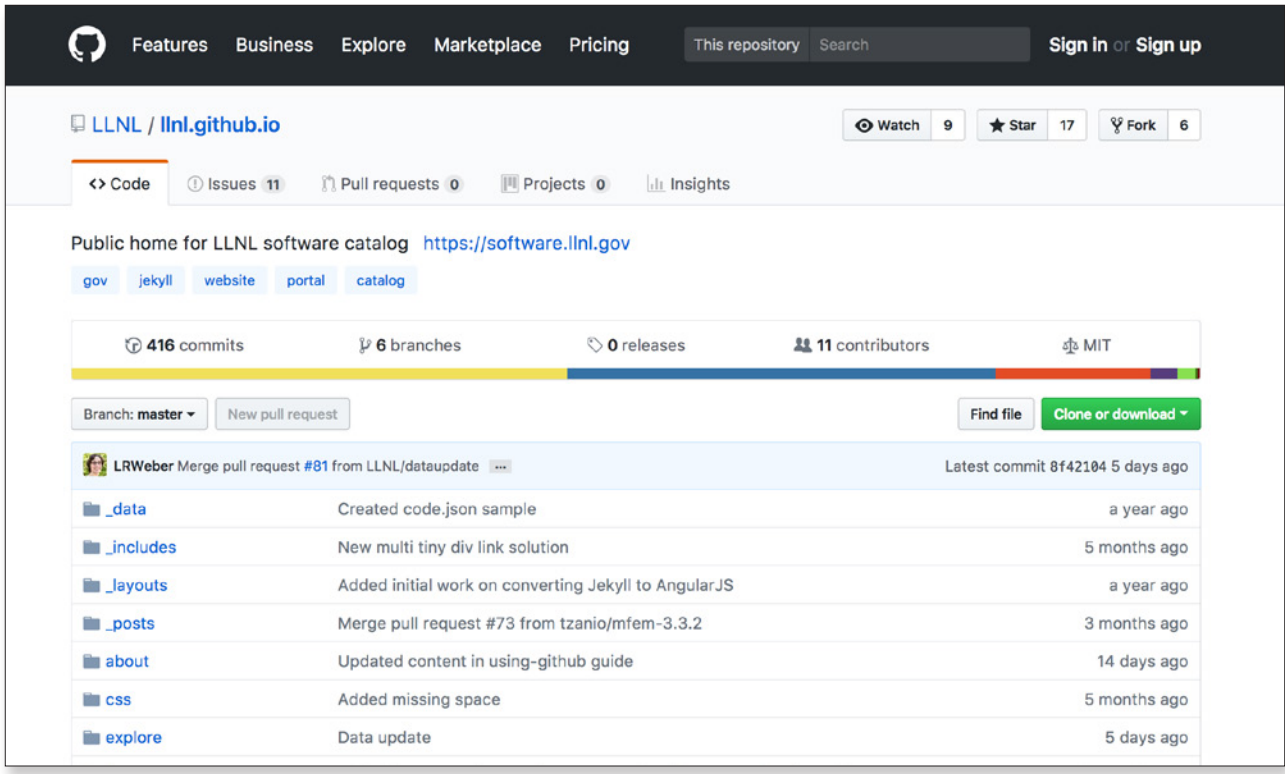
For developers, OSS participation is evidence of professional talent and productivity. "Just as a scientist can publish research papers, a developer can demonstrate his or her work through a software portfolio," explains Lee. Kolev adds, "The Laboratory's encouragement of OSS is amazing. Intellectual freedom is crucial to scientists." In addition, the open-source community gives developers a means of honing skills and provides an avenue for the Laboratory to evaluate potential hires through their contributions to Livermore and other open-source code.

Aiming for Maximum Exposure

Although Livermore has been producing OSS for many years, until recently users had no central location for accessing the Laboratory's code. Software repositories were posted to websites on the llnl.gov domain, personal websites, and numerous external hosting platforms such as SourceForge; Bitbucket; or GitHub, today's most popular source code hosting service in the world. To make Livermore OSS more easily accessible, Lee and colleagues launched a website (software.llnl.gov) in 2015. He says, "This portal offers a full corpus of what the Laboratory develops as open source by providing a centralized collection of pointers to externally hosted repositories."

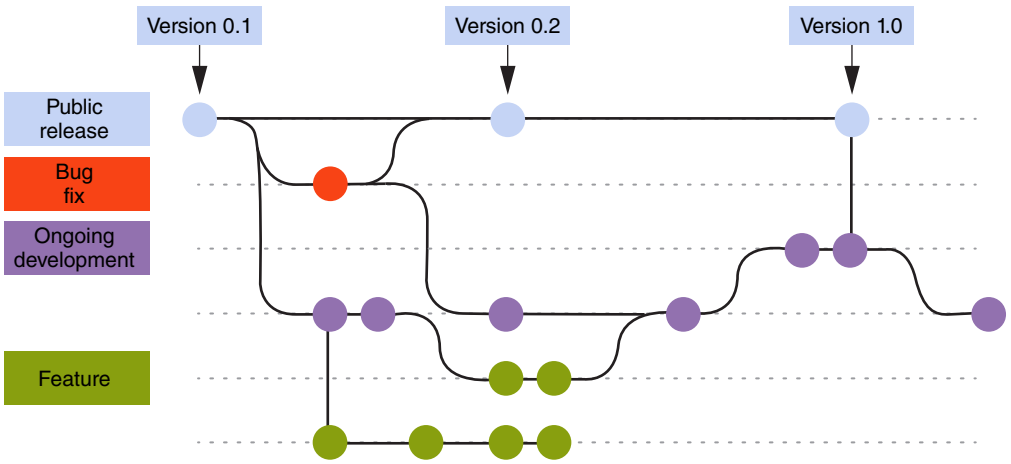
Laboratory developers create a repository for each open-source project predominantly via the Livermore organization's main page on GitHub (github.com/llnl), from which Livermore's portal dynamically pulls the information. In both locations, users can find software by name or key words. "Together, these resources facilitate the management, support, and discovery of our open-source repositories," says Lee.

On GitHub, Laboratory developers publish updates to source code and associated documentation while interacting with user communities. These users can download the software, suggest features



Livermore's open-source software (OSS) is predominantly released to the public on the GitHub hosting platform. GitHub tracks user actions (top right) such as the number of Watchers (followers who receive change notifications), Stars (signifying both bookmarking and appreciation), and Forks (copies created for independent development) for each software project.

and enhancements, report bugs, leave comments or questions, and communicate directly with Livermore developers. The repositories record each code change (known as a commit), version history, and contributor activity. Among Livermore's most actively developed and widely used software repositories are ROSE, MFEM (Modular Finite Element Methods), ZFS on Linux, and Spack.



New versions of software include a combination of enhancements, optimizations, and bug fixes. Developers releasing code to the open-source community rely on hosting platforms such as GitHub to help track and assess internal and external contributions to these new versions. (Image courtesy of www.atlassian.com. Licensed with CC BY 2.5 AU.)

A More Sophisticated Compiler

Software can be analyzed for debugging, performance tuning, and other optimization tasks. To conduct these analyses, developers depend on advanced software packages called compilers to translate human-readable code into machine-friendly binary instructions. Enter ROSE, Livermore's homegrown compiler infrastructure, which remains unique among compiler solutions after nearly 25 years of development and 14 years as OSS. Project leader Daniel Quinlan notes, "Early on, we made the argument to the Department of Energy that ROSE should be OSS because the software fulfills an ongoing need. Software is produced every year, in a range of languages, for everything from refrigerators to pacemakers to cars."

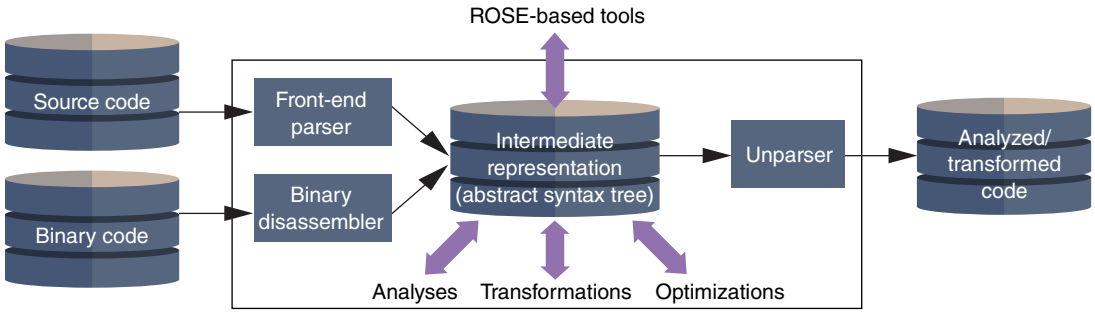
While most compilers transform source code into binary code, ROSE also generates more sophisticated source code. This source-to-source capability, combined with support for C++ and its many versions, gives researchers flexibility when optimizing simulation codes for multiple operating systems and hardware architectures. In 2009, Quinlan's team won an R&D 100 Award for this innovative technology (see *S&TR*, October/November 2009, pp. 12–13). Today, as Livermore and other institutions prepare for the exascale computing era—faster, more powerful machines working on diverse architectures—scientists use ROSE to evaluate their software's portability. "ROSE's automated transformation helps ensure codes will keep working when ported to next-generation systems," explains Quinlan.

Livermore initially released ROSE as OSS so collaborators at Argonne National Laboratory could use it, and the open-source approach has benefited the project's progress and longevity. "If developers want their work to be influential, users must be able to find it, and the barrier to entry must be low. We're always striving to make ROSE easy to use and install with good documentation," says Quinlan. Though most code contributions are made by Livermore developers, ROSE is widely used by an external community including students, researchers at other national laboratories, and project collaborators. The project has adapted to a changing market. New team members include business manager Greg Pope, who joined the group to help scale ROSE's capabilities.

Quinlan and Pope envision a bright future for ROSE as software requirements evolve for home automation and security, medical devices, financial transactions, power grids, and antiquated systems in need of coding revisions. "ROSE can evaluate software in these applications, and its automation in upgrading codes can remove some risk by reducing the probability of human error," states Pope. "Our challenge is to meet these needs without degrading technology—to balance agility and discipline."

Finite Elements, Infinite Possibilities

Downloaded from more than 80 countries at a rate of 10 downloads per day, MFEM has made an impression on



The ROSE compiler infrastructure can parse both source code and binary code. Ancillary tools developed by the Livermore team with community support perform automated analysis, transformation, and optimization tasks.

a specialized field. Project leader Kolev notes, "The field of scientific software is small but relatively crowded. MFEM has become known for high performance and flexibility." His team first released MFEM as OSS in 2010, moving the repository to GitHub in 2015.

MFEM is a discretization library for simulation codes, acting as a mathematical base layer for large-scale physics applications. Research scientists leverage MFEM's high-order finite element meshes, spaces, and discretization algorithms as building blocks for simulating and visualizing physical phenomena. By breaking down calculations into discrete components, MFEM saves application developers time and effort. "MFEM produces accurate results quickly," states Kolev, citing as frequent customers the Livermore teams who develop finite element codes such as BLAST (see *S&TR*, September 2016, pp. 4–11).

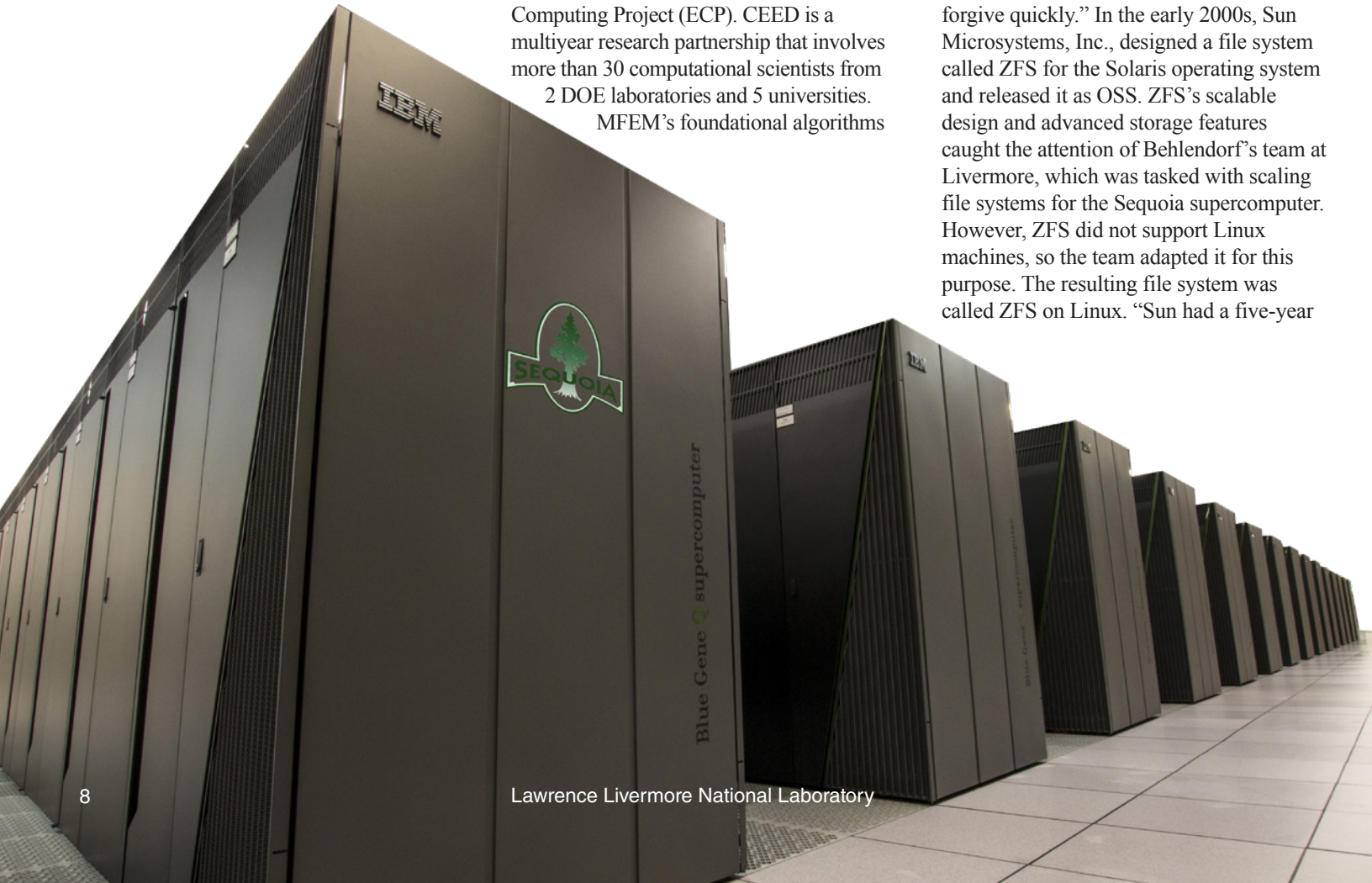
Since its debut as OSS, MFEM has evolved to include adaptive mesh refinement—a process that continuously fine-tunes a simulation's grid points—and boasts efficient performance on supercomputers. Beyond Livermore, collaborators hail from national laboratories, universities, commercial companies, and startups. "Releasing MFEM as open source has enabled us to collaborate with partners in a more seamless way. We see new capabilities made possible through their contributions," explains Kolev. "When a code is published to the user community, developers learn how others use it, making it easier to improve and uncover the code's bugs."

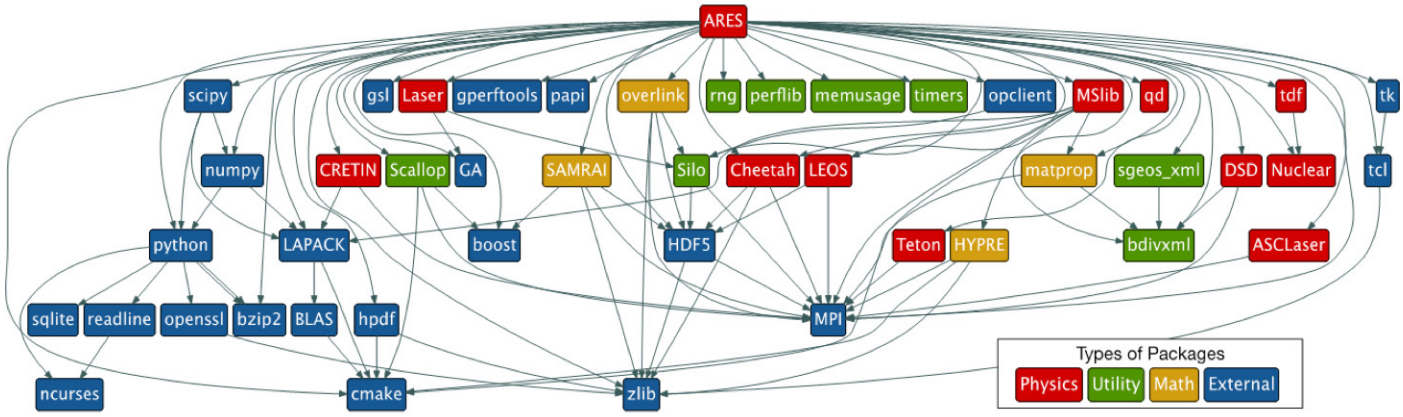
Kolev leads the Center for Efficient Exascale Discretizations (CEED), a co-design center within DOE's Exascale Computing Project (ECP). CEED is a multiyear research partnership that involves more than 30 computational scientists from 2 DOE laboratories and 5 universities. MFEM's foundational algorithms apply to several CEED objectives, such as developing high-performance simulations using high-order methods on quadrilateral, tetrahedral, and hexahedral grids. Kolev notes, "MFEM has progressed as an open-source project. Being chosen for CEED is a testament to its maturity." In addition to MFEM and other associated OSS, CEED's public forum, benchmarks, and mini-apps operate as open-source repositories on GitHub, where participants can track issues and communicate easily across all projects. "Open-source capabilities provide the only way to manage such a multi-institutional effort. We share work, offer feedback, and improve applications in a productive way," says Kolev. "The best solutions come from this approach."

Designed to Scale

According to computer scientist Brian Behlendorf, "HPC file systems are notoriously long-term investments when developed from scratch. Even a small bug can result in data loss, which users do not forgive quickly." In the early 2000s, Sun Microsystems, Inc., designed a file system called ZFS for the Solaris operating system and released it as OSS. ZFS's scalable design and advanced storage features caught the attention of Behlendorf's team at Livermore, which was tasked with scaling file systems for the Sequoia supercomputer. However, ZFS did not support Linux machines, so the team adapted it for this purpose. The resulting file system was called ZFS on Linux. "Sun had a five-year

ZFS on Linux, now OSS, was developed to create a more cost-effective, less complex, and higher performance file system for the Sequoia supercomputer.





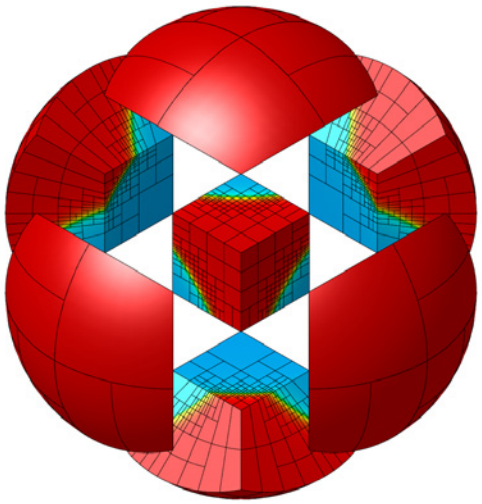
ARES, one of the Laboratory's proprietary hydrodynamics codes, has dozens of dependencies requiring a variety of package types. Spack enables multiple versions to be built nightly across different environments, reducing the code's deployment time on new machines from weeks to days.

head start on development and testing. We were able to take the next step because of their critical insights with the ZFS design—and because it was open source," explains Behlendorf.

Livermore released ZFS on Linux as OSS in 2011 and has since built ancillary tools that facilitate testing, configuration, and operating system compatibility. The file system's capabilities have expanded to include failover protection, richer accounting and quota functionality, and performance improvements that guarantee data integrity even as computing power increases. "Everything is more challenging with large-scale data," says Behlendorf. "We anticipated this challenge with Sequoia and are well situated for next-generation machines such as Sierra. ZFS on Linux was designed to scale." (See *S&TR*, March 2017, pp 4–11.)

ZFS on Linux has been a boon for companies that build data centers, institutions that store simulation or raw experimental data, and other national laboratories. "We receive contributions from many sources, even the occasional 'drive-by' contributor using it on a home system," states Behlendorf. The community's enthusiastic response has also improved Livermore's ongoing development. He continues, "We've benefited tremendously from testing by our broad range of users. They often stress the software in unique ways, which can help uncover bugs." ZFS on Linux's high quality and powerful features have prompted mainstream Linux operating systems such as Ubuntu and Debian to include the software in their distributions.

Maintaining high-quality OSS is not easy. Users' priorities may conflict with those of the project, and some contributors may not consider the amount of work required to support a software feature long term. Behlendorf cautions, "Keeping a higher level view of all needs is essential. Developers have to think outside of their own use case so they do not inadvertently break one feature by adding another." Still, the challenges are worth the effort. "One strength of OSS is that the development cost is spread out. Developers can create code much faster, and of better quality, if they are willing to share their code," he says.



The logo for the MFEM project illustrates the high-order mesh elements and physics field representations it applies to high-performance computer simulations. This image was created with GLVis, another Livermore-developed OSS that uses MFEM to generate accurate finite-element visualizations.

Recipe for Success

Large-scale scientific applications, such as mathematical and physics codes, rely on hundreds of external software programs, or packages. Installing different package versions and configurations quickly becomes laborious—developers frequently have to rebuild packages because of bug fixes, operating system upgrades, and other circumstances. As is the case with most developers, Gamblin performed this work manually until he reached his limit in 2013. "I wanted to solve the problem rather than repeat it," he says.

Gamblin created Spack, a Python-based package manager, to install software automatically. Spack operates on a range of HPC platforms because it understands the complex dependencies among HPC software packages. Spack makes it easy to leverage others' software packages, which allows application scientists to focus on science instead of software infrastructure. "Building software on a supercomputer can be a painful process," explains Gamblin.

With Spack, a research team can build many different versions of their code and test new configurations before deployment to a production environment.

Gamblin's team released Spack as OSS in 2014. Each new package added to the code base makes Spack more flexible for users working with diverse platforms, programming languages, simulation frameworks, and other variables. Saving time and effort improves reproducibility and performance. "Spack is a repository of recipes for building HPC software. It lets users leverage a larger software ecosystem because they are not spending their time rediscovering how to build every package," says Gamblin.

Spack's 2016 average of 100 downloads per day ballooned to 400 downloads per day in 2017. Every week, the GitHub repository logs 7,000 views while the team assesses hundreds of contributions. They

Livermore computer scientist Todd Gamblin delivers Spack tutorials, talks, and presentations at supercomputing conferences and other face-to-face meetings. Participating in industry events allows Laboratory scientists to connect and engage with the open-source community. (Photo by Meg Epperly.)

External Software Enhances Internal Programs

Lawrence Livermore scientists are often contributors to and customers of externally developed open-source software (OSS), including frameworks such as Python and Drupal, which underpin programmatic work. Computer engineer Ian Lee notes, "We can lead the charge in user communities where a national laboratory may not normally have a seat at the table but where our needs must be discussed."

At the Laboratory's National Atmospheric Release Advisory Center (NARAC), a Livermore software development team is using open-source technologies to modernize NARAC's Linux-based central system. NARAC operations depend on a fast, reliable computer modeling system to provide real-time assessment of emergency response strategies for airborne releases of hazardous materials (see *S&TR*, January/February 2012, pp. 12–18). Led by information technology and software manager John Fisher, the multiphase modernization effort includes rebuilding the central system's graphical user interface (GUI) framework with a scalable OSS suite. "We are implementing software for the long term, so we have chosen OSS with broad community support," says Fisher.

The central system consists of 50 data sources, 28 servers, 8 storage units, and more than 3 million lines of code. Fisher's team explored OSS that could simplify this large, complex architecture and selected the popular Angular platform developed by Google. According to software developer Sei Jung Kim, the decision to use Angular informed subsequent decisions. "Angular recently underwent a major redesign. We pursued solutions for the visual layer that were compatible with the platform," she says. For instance, the team turned to the PrimeNG user interface toolkit to provide a consistent, attractive, and powerful user experience.

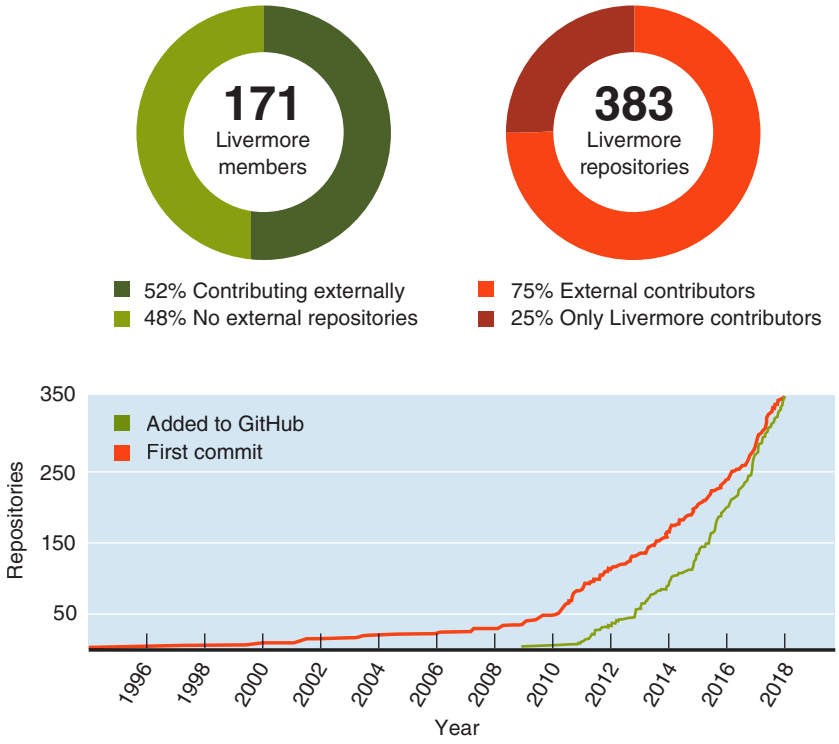
One challenge of combining open-source tools is determining which versions to use. For example, Kim states, "Every time we update PrimeNG to the latest version, we must ensure other dependencies, such as Angular, are in sync." In addition, the team must work through compatibility issues among open-source solutions. "When you use multiple tools, they don't always like each other," notes Fisher. Every OSS integrated into the system undergoes a detailed evaluation by NARAC developers.

NARAC's new technology stack is designed to streamline development while delivering a better user experience and avoiding software obsolescence. Fisher remarks, "The maturity of our current system allows us to more easily identify the capabilities we need. Finding well-supported, well-documented solutions for niche requirements is work, but not using open-source tools makes development and maintenance more difficult and expensive in the long run." Already the team is seeing usability improvements with several desktop applications developed from OSS. For example, an observed meteorological database GUI allows NARAC users to browse real-time global weather data such as temperature, humidity, wind speed, and air pressure. Around 10 gigabytes of data accumulate per day, but the user only wants information related to a specific event under analysis. "The GUI can bring up a selected portion of data based on user input," explains Kim.

As the central system modernization project progresses, Fisher remains confident in the team's approach to reducing dependency on proprietary software. "The Laboratory's work demands sophisticated technology solutions," he says. "With open-source tools, we are upgrading existing functionalities and adding new capabilities with better usability."

have adapted to this growth with cloud-based testing services and continuous integration, a process that merges multiple copies of code to detect any build issues. "One challenge is maintaining stability while updating features incrementally. However, Spack wouldn't be as stable as it is if users didn't find problems with it," notes Gamblin.

Although Spack can run on personal computers or relatively small clusters, significant contributions come from users at other HPC centers. Livermore's relationship with the community is a win–win. Gamblin explains, "External users have contributed most of Spack's more than 2,400 packages, which Livermore could not have developed alone." This success has led others to take notice, especially for efforts such as DOE's ECP, which has adopted Spack to manage software releases for DOE's entire exascale software stack. "Package managers are the glue that hold software ecosystems together because they allow developers to use each others' software with a push of a button," says Gamblin. "We are optimistic that Spack will be that glue for ECP and enable a thriving exascale software ecosystem."



Livermore's OSS portal tracks software release trends in dynamic graphs developed by computer scientist Laura Weber. (top left) More than half of Livermore developers contribute to externally developed OSS. (top right) Three-fourths of Livermore open-source projects receive external contributions. (bottom) The first Laboratory-developed open-source repositories were created years before Git (2005) and GitHub (2008) emerged as community-based solutions.

Paying It Forward

Livermore's OSS portfolio has grown thanks to an encouraging culture at the Laboratory. For instance, the site-wide Developer Day session gives employees across the Laboratory a chance to learn about each other's projects, including OSS efforts. At Livermore's seasonal "hackathons," developers try out OSS with existing work or new projects under consideration (see *S&TR*, June 2013, pp. 16–18). This commitment, in turn, has increased Livermore's presence and leadership in the open-source community.

Beyond online interaction, Laboratory scientists actively connect with the open-source community by attending industry events and presenting at conferences. "When the software we describe at these events is open source, the audience can access and examine it right away, which helps convince them that it's worth their time to investigate further," states Kolev. Lee presented at the 2016 GitHub Universe Conference. Gamblin gives Spack tutorials and presentations at supercomputing conferences, and Behlendorf is a fixture at OpenZFS events. Lee and Gamblin also run Twitter accounts (@LLNL_OpenSource and @spackpm) to publicize the Laboratory's OSS news and communicate with those interested in Spack and other projects.

Livermore's software development workforce embodies this collaborative spirit, and Lee strongly encourages sharing code as the default in cases where classification and sensitivity are not issues. "Embracing an open-source strategy means we can make code developed by one available to all. Sharing information with others working in the same technology stack and receiving their constructive feedback is quite valuable," he advises. "Without this feedback along the way, more rework would be needed later. Code does not have to be perfect to be useful."

—Holly Auten

Key Words: Center for Efficient Exascale Discretizations (CEED), compiler, data storage, discretization, Exascale Computing Project (ECP), Federal Source Code Policy, file system, GitHub, high-performance computing (HPC), MFEM (Modular Finite Element Methods), National Atmospheric Release Advisory Center (NARAC), open-source software (OSS), package manager, portal, repository, ROSE, source code, Spack, ZFS on Linux.

For more information contact Ian Lee (925) 423-4941 (lee1001@llnl.gov).